V850ES/Jx3 Microcontrollers Usage Restrictions	Document No.	ZBG-CC-09-0047	1/4
	Date issued	November 9, 2009	
	Issued by	Product Solution Group Multipurpose Microcomputer Systems Division Microcomputer Operations Unit NEC Electronics Corporation	

Related documents: V850ES/JG3 Hardware User's Manual: U18708EJ2V0UD00 (2nd edition) V850ES/JJ3 Hardware User's Manual: U18376EJ3V0UD00 (3rd edition) V850ES Architecture User's Manual: U15943EJ3V0UM00 (3rd edition)	Notification classification	√	Usage restriction
			Upgrade
			Document modification
			Other notification

1. Affected products

All V850ES/Jx3 microcontrollers

V850ES/JG3: μPD70F3739, μPD70F3740, μPD70F3741, μPD70F3742

V850ES/JJ3: μPD70F3743, μPD70F3744, μPD70F3745, μPD70F3746

2. Notification

The following restriction for the V850ES/Jx3 microcontrollers has been found.

No. 1 Restriction on executing a `mul` or `mulu` instruction

Description:

The following occur if a specific instruction sequence (sequence 1 or 2 below) is executed.

- The result of executing a multiplication instruction is not stored in the relevant general-purpose register.
- As a result of executing an `ld` instruction for a mis-aligned address, the data at an incorrect address is read and stored in the relevant general-purpose register.

Sequence 1:

In the following instruction sequence, the RAM is read by one of the instructions in (2) at the same time as the RAM is accessed by a DMA transfer:

(1) `ld` or `sld`: A load instruction for the internal ROM

(2) `ld` or `sld`: A load instruction for the internal RAM

(3) `mul` or `mulu`: An instruction that multiplies word data and whose result is truncated to 32 bits[Note 1]

... **Note 2**

(4) `ld` or `sld`: A load instruction for a mis-aligned address in the internal ROM or RAM

Notes 1. For a `mul` or `mulu` instruction, the operation described in this restriction occurs if `r0` is specified for the third operand (`reg3`), or the same register is specified for the second operand (`reg2`) and third operand (`reg3`), as shown below:

`mul reg1, reg2, reg3` (`reg3` = `r0` or `reg2` = `reg3`)

`mul imm9, reg2, reg3` (`reg3` = `r0` or `reg2` = `reg3`)

`mulu reg1, reg2, reg3` (`reg3` = `r0` or `reg2` = `reg3`)

`mulu imm9, reg2, reg3` (`reg3` = `r0` or `reg2` = `reg3`)

For a program written in C, the CA850 does not generate the `mul` and `mulu` instructions if the same register is specified for the `reg2` and `reg3` operands.

2. This restriction applies if it takes 2 clock cycles or less between the instructions in (3) and (4).

This restriction does not apply in any of the following cases:

- DMA is not used to transfer data to or from the internal RAM.
- The data read by the load instruction in (1) is used to specify the target address of the load instruction in (2).
- The data read by the load instruction in (2) is referenced by the multiplication instruction in (3).
- The data obtained by the multiplication instruction in (3) is used to specify the target address of the load instruction in (4).
- The data obtained by the multiplication instruction in (3) is referenced by an instruction executed between the instructions in (3) and (4).
- At least one of the following instructions is executed between the instructions in (3) and (4):
  - A multiplication instruction (`mul`, `mulh`, `mulhi`, `mulu`)
  - A bit manipulation instruction (`clr1`, `not1`, `set1`, `tst1`)
  - A special instruction (`callt`, `dispose`, `switch`)
- The instruction in (4) is a load instruction that accesses the memory in bytes (`ld.b`, `ld.bu`, `sld.b`, or `sld.bu`).
- The instructions in (1) to (4) are located in an external memory or the internal RAM.

Sequence 2:

In the following instruction sequence, access by the instruction in (1) ends at the same time as the instruction in (2) accesses the internal RAM:

(1) `ld` or `sld`: A load instruction for an external memory

... **Note 1**

(2) `ld` or `sld`: A load instruction for the internal RAM

(3) `mul` or `mulu`: An instruction that multiplies word data and whose result is truncated to 32 bits[Note 2]

... **Note 3**

(4) `ld` or `sld`: A load instruction for a mis-aligned address in the internal ROM or RAM

Notes 1. This restriction applies if a different instruction is executed between the load instructions in (1) and (2), and the access by the instruction in (1) ends at the same time as the instruction in (2) accesses the internal RAM.

2. For a `mul` or `mulu` instruction, the operation described in this restriction occurs if `r0` is specified for the third operand (`reg3`), or the same register is specified for the second operand (`reg2`) and third operand (`reg3`), as shown below:

`mul reg1, reg2, reg3` (`reg3` = `r0` or `reg2` = `reg3`)

`mul imm9, reg2, reg3` (`reg3` = `r0` or `reg2` = `reg3`)

`mulu reg1, reg2, reg3` (`reg3` = `r0` or `reg2` = `reg3`)

`mulu imm9, reg2, reg3` (`reg3` = `r0` or `reg2` = `reg3`)

For a program written in C, the CA850 does not generate the `mul` and `mulu` instructions if the same register is specified for the `reg2` and `reg3` operands.

3. This restriction applies if it takes 2 clock cycles or less between the instructions in (3) and (4).

This restriction does not apply in any of the following cases:

- An external bus is not used.
- The data read by the load instruction in (1) is used to specify the target address of the load instruction in (2).
- The data read by the load instruction in (1) is referenced by an instruction executed between the instructions in (1) and (2).
- The data read by the load instruction in (2) is referenced by the multiplication instruction in (3).
- The data obtained by the multiplication instruction in (3) is used to specify the target address of the load instruction in (4).
- The data obtained by the multiplication instruction in (3) is referenced by an instruction executed between the instructions in (3) and (4).
- At least one of the following instructions is executed between the instructions in (3) and (4):
  - A multiplication instruction (`mul`, `mulh`, `mulhi`, `mulu`)
  - A bit manipulation instruction (`clr1`, `not1`, `set1`, `tst1`)
  - A special instruction (`callt`, `dispose`, `switch`)
- The instruction in (4) is a load instruction that accesses the memory in bytes (`ld.b`, `ld.bu`, `sld.b`, or `sld.bu`).
- The instructions in (1) to (4) are located in an external memory or the internal RAM.

Workaround:

Action for systems being developed and to be developed in future:

We regard this as a restriction related to the CPU's features and do not intend to correct the microcontroller. Instead, the CA850 compiler will be modified to automatically prevent instructions to which this restriction applies from being generated. This workaround does not work for instructions in assembly code, so the CA850 outputs a message for such instructions. How NEC Electronics will provide the update for the compiler is shown below.

If you are using another compiler, contact an NEC Electronics sales representative.

- If your compiler is the NEC Electronics compiler CA850:

  The CA850 will be upgraded to add the above workaround.

  The following table shows the version and release schedule for the C compiler package CA850, and the software package SP850 that includes the CA850.

Product Name	Current CA850 Version	Version of CA850 After Upgrade	Language	Release Schedule
SP850, CA850	V3.40	V3.42	Japanese	November 9, 2009
			English	November 9, 2009

Action for already-developed systems:

Use the interview sheet in attachment 2 to check whether the restriction applies.

Application of this restriction to embedded software products:

This restriction applies to NEC Electronics real-time OSs and middleware as follows:

- Real-time OSs:

  RX850: The restriction does not apply.

  RX850 Pro: The restriction does not apply.

  RX850V4: The restriction does not apply.

- Middleware:

  GOFAST: The restriction does not apply.

  JPEG: The restriction does not apply.

For products other than the above, contact an NEC Electronics sales representative. For third-party products, contact the vendor of the product.

Modification:

The device will not be corrected, so regard this restriction as a specification.

Use the latest compiler to avoid this restriction.

3. Development environment required to work around this restriction

Use NEC Electronics compiler CA850 V3.42 or later.

Note that the outputs from the above version might differ from the outputs from the older versions.

If you are using another compiler, contact an NEC Electronics sales representative.

4. Document revision history

Document Number	Issued on	Description
ZBG-CC-09-0047	November 9, 2009	1st edition. Restriction no. 1

V850ES/Jx3 Microcontroller Usage Restrictions

No.	Restriction	Nickname	Product	**Note**	Development Environment in Which the Restriction Can Be Worked Around
1	Restriction on executing a `mul` or `mulu` instruction	V850ES/ JG3	μPD70F3739	△	Category: Compiler Product name: CA850 Version: V3.42 or later
			μPD70F3740	△	
			μPD70F3741	△	
			μPD70F3742	△	
		V850ES/ JJ3	μPD70F3743	△	
			μPD70F3744	△	
			μPD70F3745	△	
			μPD70F3746	△	

Note Whether or not the restriction applies

Remark The meaning of each symbol for **Note** is as follows:

−: Restriction does not apply
○: Restriction is already corrected
×: Restriction applies (correction is planned)
△: Restriction applies (correction is not planned)

• First judgment: Judgment based on product usage conditions

Check the conditions under which you are using the product to see whether the restriction applies to the product. If the restriction might apply, perform a second judgment. If the restriction is judged to be not applicable, subsequent checking is not necessary.

■ Checking the usage conditions

Select **Yes** or **No** for whether the features in (1) and (2) below are used. If there is no item for which **Yes** is selected, the restriction does not apply.

(1) Data is not transferred to or from the internal RAM using DMA.

☐ Yes ☐ No

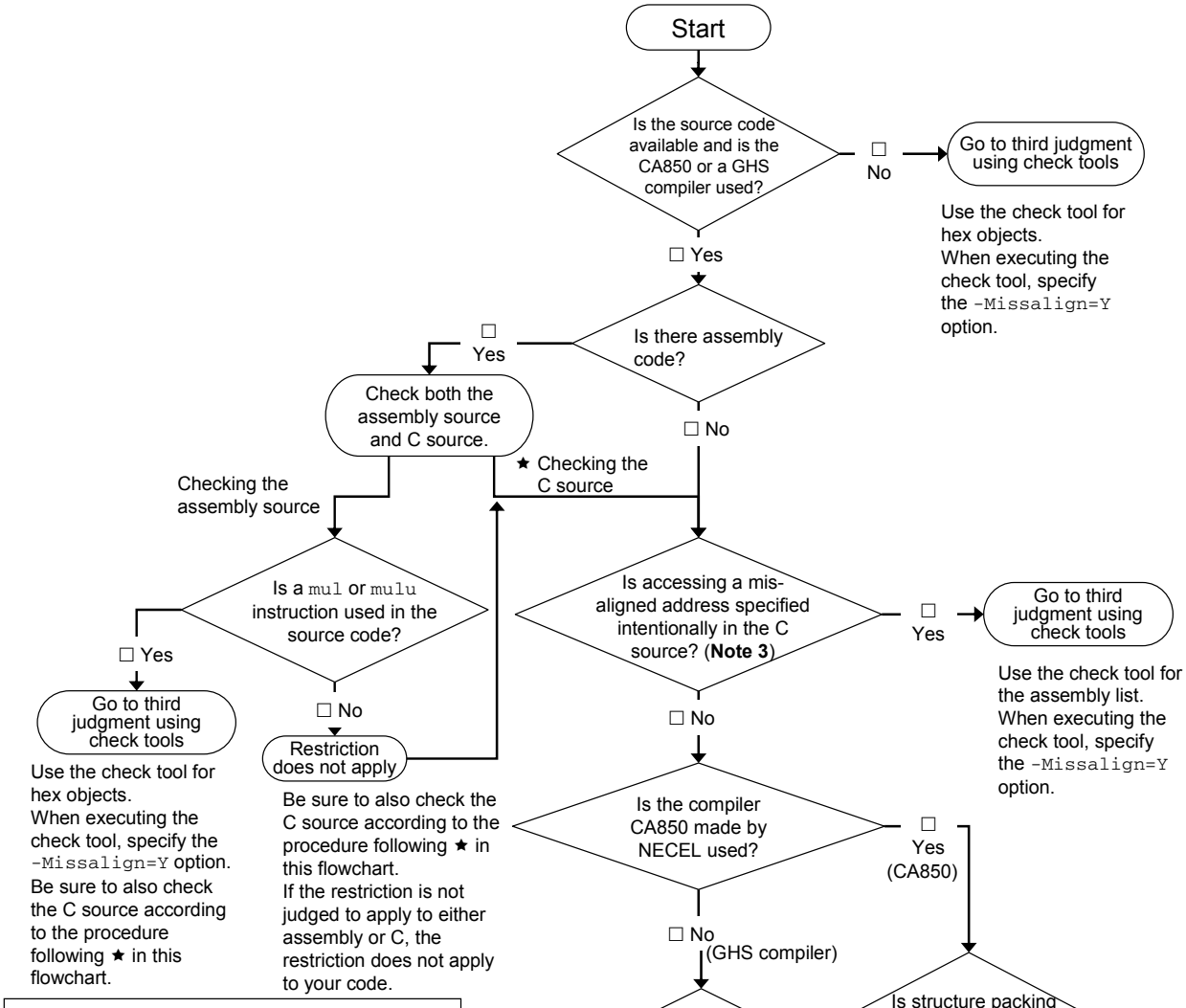
(2) An external bus interface is used.

☐ Yes ☐ No

- **Second judgment: Judgment based on compiling conditions**

Interview Sheet (for second judgment)

Use the following flowchart to judge whether the restriction applies. If the restriction might apply, a third judgment using check tools is required.


Start
Is the source code available and is the CA850 or a GHS compiler used?
☐ No
Go to third judgment using check tools
Use the check tool for hex objects.
When executing the check tool, specify the -Missalign=Y option.
☐ Yes
Is there assembly code?
☐ Yes
Check both the assembly source and C source.
☐ No
★ Checking the C source
Checking the assembly source
Is a mul or mulu instruction used in the source code?
☐ Yes
Go to third judgment using check tools
Use the check tool for hex objects.
When executing the check tool, specify the -Missalign=Y option.
Be sure to also check the C source according to the procedure following ★ in this flowchart.
☐ No
Restriction does not apply
Be sure to also check the C source according to the procedure following ★ in this flowchart.
If the restriction is not judged to apply to either assembly or C, the restriction does not apply to your code.
Is accessing a mis-aligned address specified intentionally in the C source? (Note 3)
☐ Yes
Go to third judgment using check tools
Use the check tool for the assembly list.
When executing the check tool, specify the -Missalign=Y option.
☐ No
Is the compiler CA850 made by NECEL used?
☐ Yes (CA850)
☐ No (GHS compiler)
Is structure packing at compile time specified (Note 2)?
☐ No
Restriction does not apply
☐ Yes
Is structure packing at compile time specified (Note 1)?
☐ No
Restriction does not apply
☐ Yes
Go to third judgment using check tools
Use the check tool for the assembly list.
When executing the check tool, specify the -Missalign=Y option.

Note 1: When using the CA850
The restriction might apply if either the following conditions is satisfied:

1. `#pragma pack(1)` or `#pragma pack(2)` is specified in the source code.

2. `-Xpack=1` or `-Xpack=2` is specified as a compiler option.

Note 2: When using a GHS compiler
The restriction might apply if either the following conditions is satisfied:

1. `#pragma pack(1)` or `#pragma pack(2)` is specified in the source code and `-misalign_pack` is specified as a compiler option.

2. `-pack=1` or `-pack=2` and `-misalign_pack` are specified as compiler options.

Note 3: A mis-aligned address might be accessed if a pointer to a `char` is cast to a pointer to an `int` to reference the `int`, as shown below.

```
func()
{
  char  *ptr_char="abcdef" ;
  int    data,  *ptr_int ;

    ptr_int= (int *)
(ptr_char+2);
   data = *ptr_int;
}
```